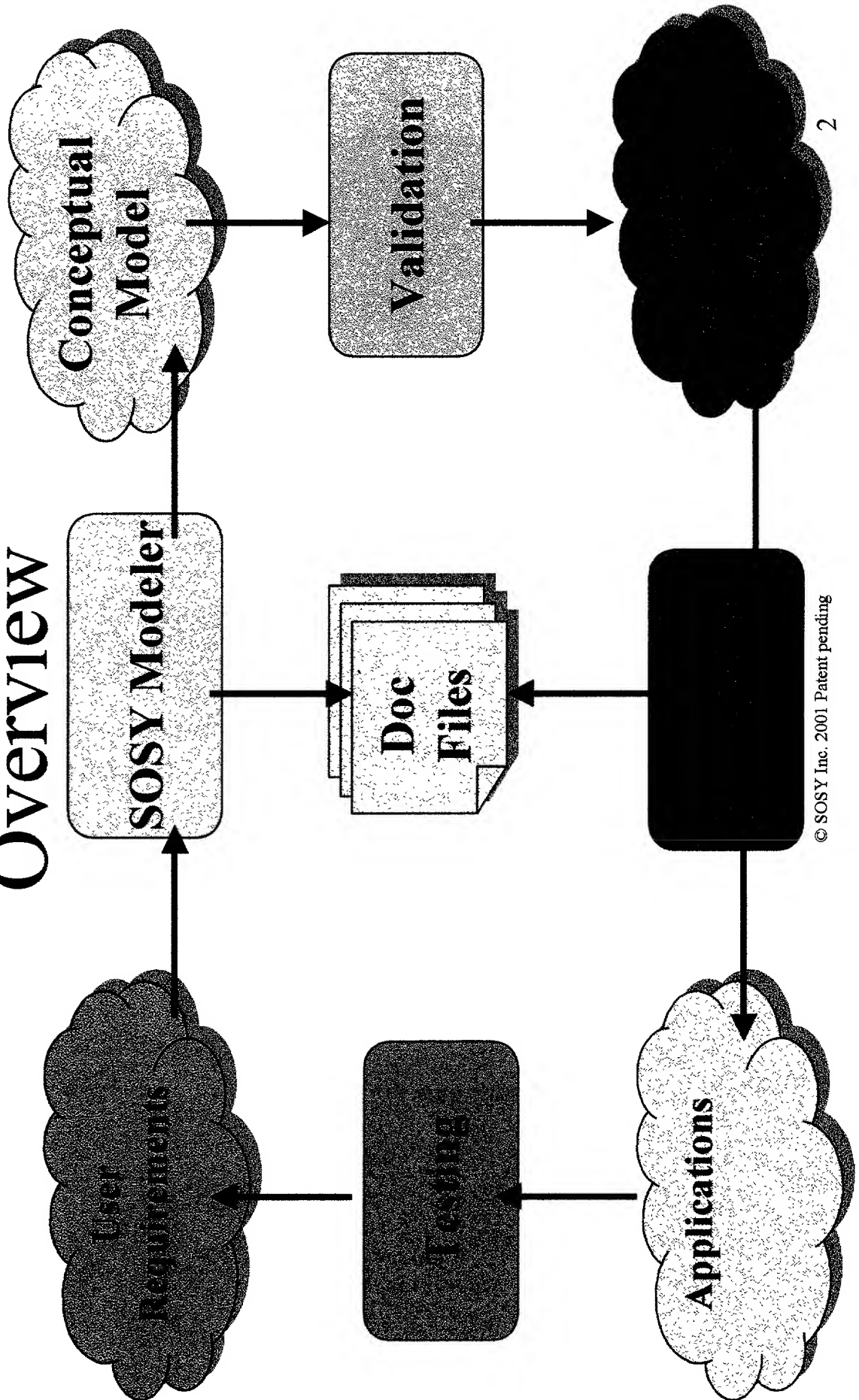


Summary

- [REDACTED]
- [REDACTED]
- [REDACTED]

Overview



Conceptual Modeling Phase

CARE Technologies, S.A.

Index

- Intro
- Overview
- Phase 0. Requirements elicitation.
- Phase 1. Classes identification.
- Phase 2. Relationships between classes.
- Phase 3. Filling classes' details.

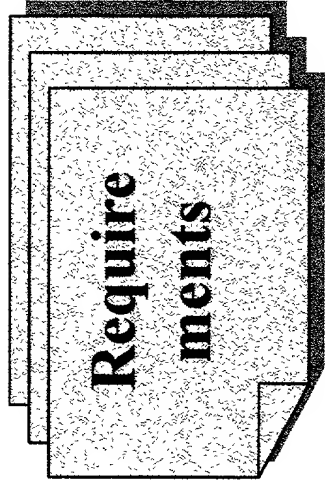
Index

- Phase 4. Express evaluations.
- Phase 5. Agent relationships.
- Phase 6. State Transition Diagram.
- Phase 7. Presentation Model.

Intro

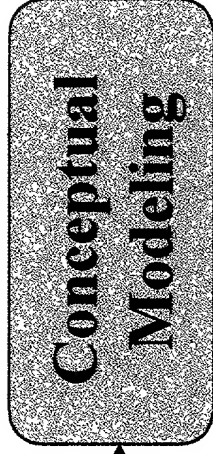
- Conceptual Modeling Phase is a process of systematically & precisely description of the system to build, using:
 - Graphical UML compliant diagrams.
 - Constrains and semantics in a formal non-ambiguous language.
 - This phase is assisted by an integrated Modeler tool.

Overview



Requirements

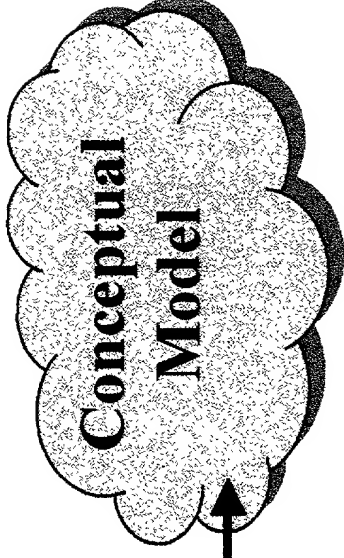
- Specifications
- Documents
- Interviews
- Reports
- Other info. sources



Conceptual Model

- Classes
- Relationships
- Attributes
- Services
- ...

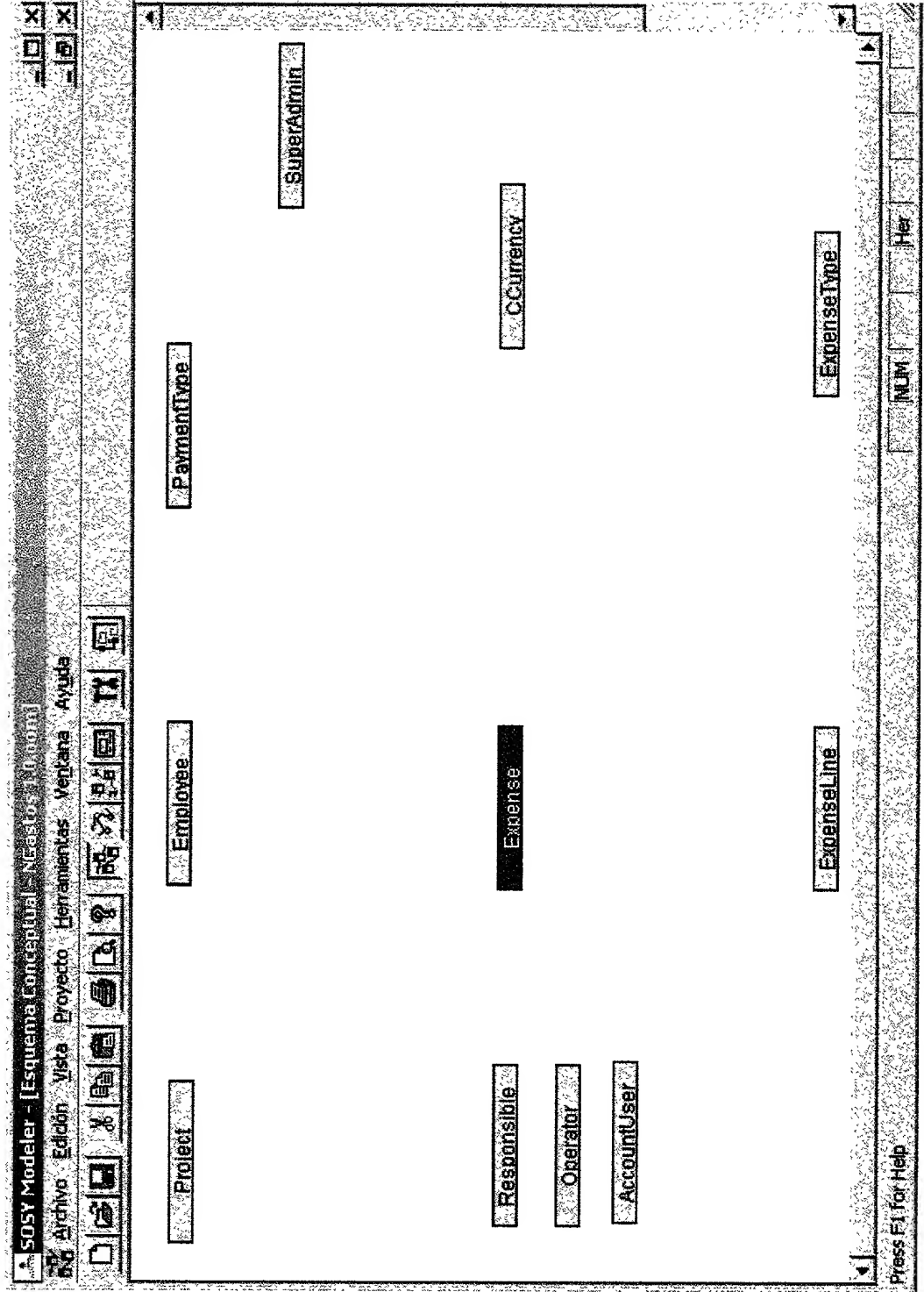
Expressed in a non-ambiguous language.



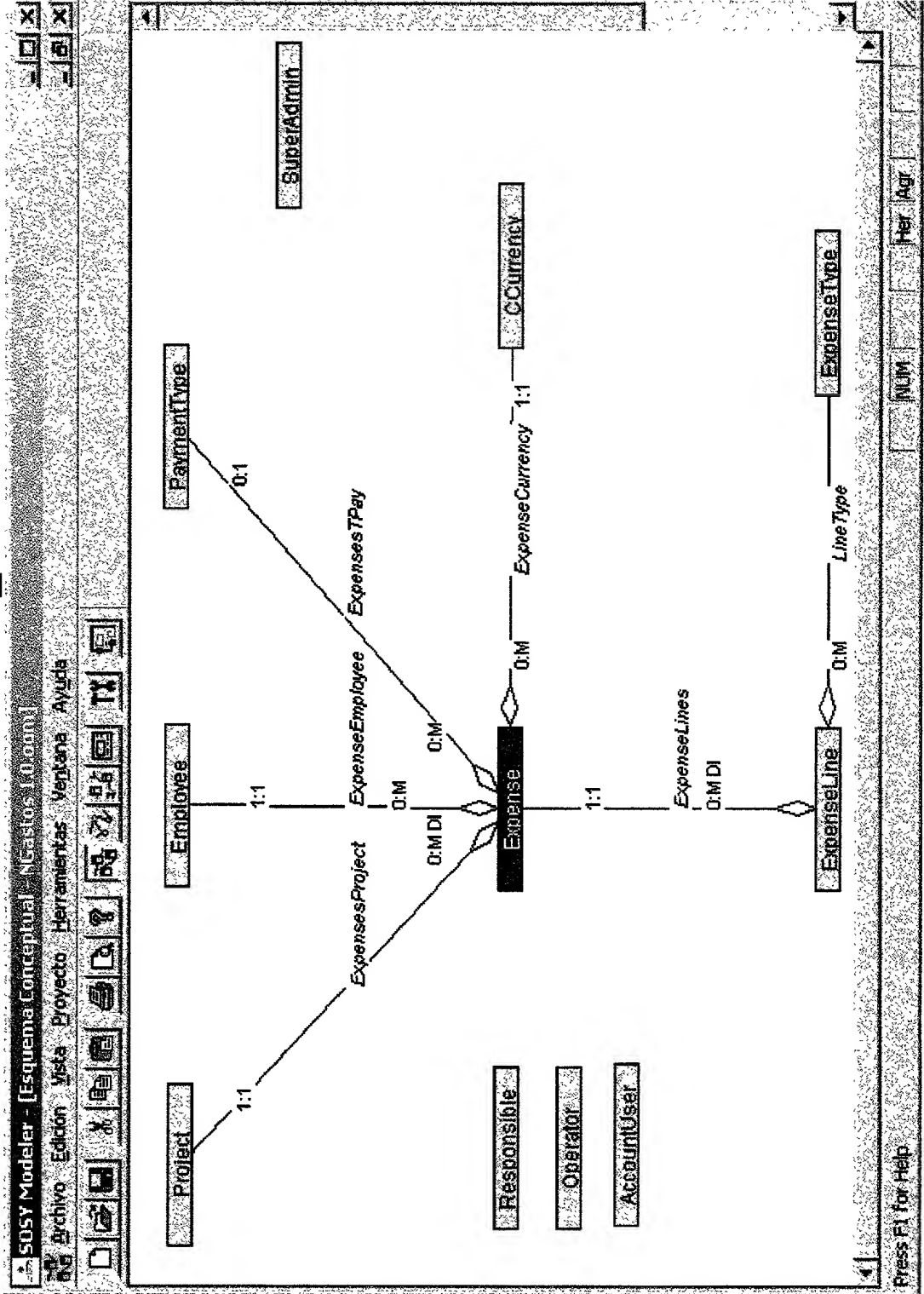
Phase 0. Requirement elicitation.

- - Gathering the system requirements.
 - By meetings & interviews with customers, experts and final users.
 - By collecting reports, or documents expressing the system how-to and using tools.
 - Obtaining a coherent set of information as input to the next phase.
-

Phase 1. Classes identification.



Phase 2. Relationships between classes.



Phase 3. Filling classes' details.

Clase

Atributos

Servicios

Derivaciones

Restricciones

Agentes

Transacciones

Relaciones

Generalidades

Atributos

Nombre	Tipo atributo	Tipo dato	Id	Tamaño	Valor defecto	Pedir al crear	Nulos	Afecta
PresentDate	Constante	Date			today()	Si	No	Afecta
Status	Variable	Int			0	No	No	Modificar
Cause	Variable	String		255		Si	No	Borrar
AuthoDate	Variable	Date			NULL	No	Si	
AuthoComments	Variable	String		255	NULL	No	Si	
PaymentDate	Variable	Date			NULL	No	Si	
PayComments	Variable	String		255	NULL	No	Si	
TotExpenses	Derivado	Real					Si	
TotExpensesCur	Derivado	Real			0	Si	No	
Advances	Variable	Real						
AdvancesCur	Derivado	Real						
Exchange	Variable	Real						
Balance	Derivado	Real					Si	
BalanceCur	Derivado	Real						

Nombre

Tipo Atributo

Tipo Dato

Limpiar

Notas >>

Alias

Observaciones

Clase

Expense

Aceptar

Cancelar

Phase 3. Filling classes' details.

Clase

Ambulos | Servicios | Derivaciones | Restricciones | Agentes | Transacciones | Relaciones | Generalidades

Atributo

Balance

Fórmulas de Derivación

Condición	Fórmula
	Total Expenses - Advances

Añadir

Modificar

Borrar

Condición

Fórmula

Observaciones

Clase: Expense

Aceptar

Cancelar

Phase 3. Filling classes' details.

Clase

Atributos | Servicios | Denegaciones | Restricciones | Agentes | Transacciones | Relaciones | Generalizables

Eventos y Transacciones

Servicios

Nombre	Características
newexpense	New
delexpense	Destroy
modify	
close	
authorize	
approve	
pay	
rejectautho	
rejectpayment	
insPaymentType	Shared with PaymentType
borPaymentType	Shared with PaymentType
DELE TEALL	Tran
TPAY	Tran

Añadir
Modificar
Borrar
Copiar

Parámetros

Nombre	Tipo dato
p_thisExpense	Expense
p_Cause	String
p_Advances	Real
p_Exchange	Real

Añadir
Modificar
Borrar

Atributos

Notes >>

Nombre:

☐ Transacción
☐ New ☐ Destroy
☐ Uso interno

Alias:

Mensaje de Ayuda:

Observaciones:

Nombre:

Tamafio:

Alias:

Valor por defecto:

Observaciones:

Nombre:

Tipo dato:

☐ Nulos
☐ Colección

Valor por defecto:

Observaciones:

Clase:

Aceptar Cancelar

Phase 3. Filling classes' details.

Clase

Atributos

Servicios

Derivaciones

Restricciones

Agentes

Transacciones

Relaciones

Generalidades

Transacción:

DELETEALL

Fórmula

[FOR ALL Lines DO Lines.deleteLine(Lines) . deleteExpense(THIS)]

Acción

Clase/Rol

Expense

Agentes

Servicio:

approve

Parámetros:

p_thisExpense

Inicializar

Crear Parámetro en la Transacción

Observaciones

Clase:

Expense

Aceptar

Cancelar

APPENDIX A

14

Phase 3. Filling classes' details.

Clase

Atributos | Servicios | Derivaciones | Restricciones | Agentes | Transacciones | Relaciones | Generalidades

Estáticas

Exchange > 0

Añadir

Modificar

Borrar

Heredadas

Fórmula

Exchange > 0

Message De Error: Exchange must be greater than zero

Dinámicas

Oper Temporal

Condicion1

Oper Temporal

Condicion2

Fórmula

Message De Error:

Añadir

Modificar

Borrar

Clase:

Expense

Aceptar

Cancelar

Phase 4. Express evaluations.

Modelo Funcional

Clase: Expense

Atributo: Cause

Acceptar

Cancelar

Evento
modify

Efecto
p_Cause

Condición

Añadir

Modificar

Borrar

Cardinal

De Estado

De Situación

Detalles de Evaluación

Evento: modify

Condición de evaluación

IF

Efecto del evento

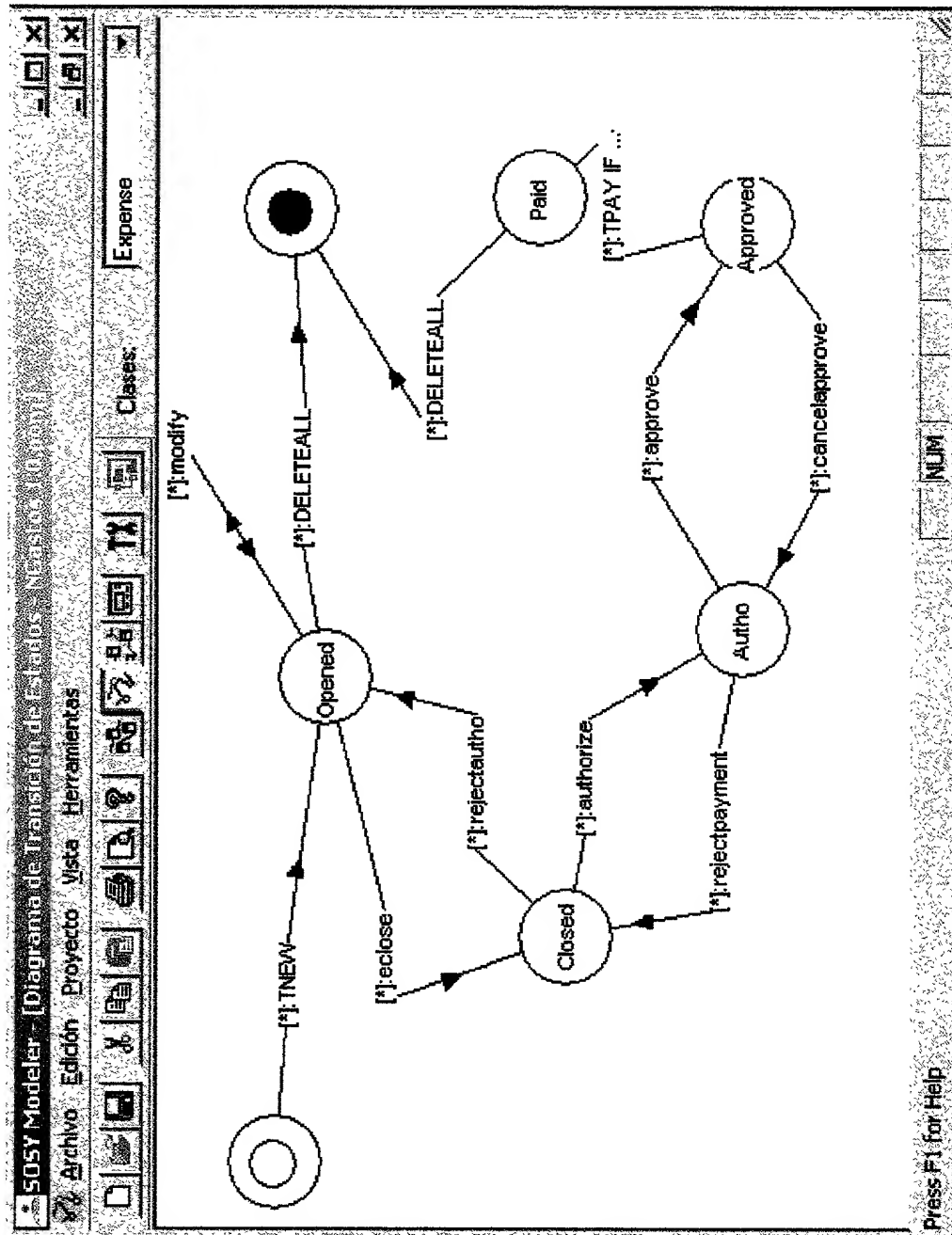
p_Cause

☐ Inferir para el resto de atributos

Phase 5. Agent relationships.

Clase																													
Atributos Servicios Derivaciones Restricciones Agentes Transacciones Relaciones Generalidades	Clase servidor: <input type="text" value="Expense"/>	<p>Servicios:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 80%;">Servicio</th><th style="width: 20%;">Temporalidad</th></tr> </thead> <tbody> <tr><td>newexpense</td><td>No</td></tr> <tr><td>modify</td><td>No</td></tr> <tr><td>close</td><td>No</td></tr> <tr><td>authorize</td><td></td></tr> <tr><td>rejectautho</td><td></td></tr> <tr><td>DELETEALL</td><td></td></tr> </tbody> </table> <div style="margin-top: 10px; text-align: center;"> Añadir >> << Eliminar Todos >> </div>	Servicio	Temporalidad	newexpense	No	modify	No	close	No	authorize		rejectautho		DELETEALL		<div style="margin-bottom: 10px;"> Hacer temporal No hacer temporal </div> <p>Clase AccountUser tiene visibilidad sobre:</p>												
Servicio	Temporalidad																												
newexpense	No																												
modify	No																												
close	No																												
authorize																													
rejectautho																													
DELETEALL																													
Clase: <input type="text" value="AccountUser"/>	Clase servidora: <input type="text" value="Expense"/>	<p>Atributos:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 80%;">Atributos</th><th style="width: 20%;">Temporalidad</th></tr> </thead> <tbody> <tr><td>Cause</td><td></td></tr> <tr><td>AuthoComments</td><td></td></tr> </tbody> </table> <div style="margin-top: 10px; text-align: center;"> Añadir >> << Eliminar Todos >> </div>	Atributos	Temporalidad	Cause		AuthoComments		<p>Atributos:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 80%;">Atributos</th><th style="width: 20%;">Temporalidad</th></tr> </thead> <tbody> <tr><td>Expense.id_Expense</td><td></td></tr> <tr><td>Expense.PresentDate</td><td></td></tr> <tr><td>Expense.Status</td><td></td></tr> <tr><td>Expense.PaymentDate</td><td></td></tr> <tr><td>Expense.PayComments</td><td></td></tr> <tr><td>Expense.AuthoDate</td><td></td></tr> <tr><td>Expense.TotExpenses</td><td></td></tr> <tr><td>Expense.TotExpensesCur</td><td></td></tr> <tr><td>Expense.Advances</td><td></td></tr> </tbody> </table> <div style="margin-top: 10px; text-align: center;"> Añadir >> << Eliminar Todos >> </div>	Atributos	Temporalidad	Expense.id_Expense		Expense.PresentDate		Expense.Status		Expense.PaymentDate		Expense.PayComments		Expense.AuthoDate		Expense.TotExpenses		Expense.TotExpensesCur		Expense.Advances	
Atributos	Temporalidad																												
Cause																													
AuthoComments																													
Atributos	Temporalidad																												
Expense.id_Expense																													
Expense.PresentDate																													
Expense.Status																													
Expense.PaymentDate																													
Expense.PayComments																													
Expense.AuthoDate																													
Expense.TotExpenses																													
Expense.TotExpensesCur																													
Expense.Advances																													
		<div style="display: flex; justify-content: space-between; width: 100%;"> Aceptar Cancelar </div>																											

Phase 6. State Transition Diagram.



Phase 6. STD Preconditions

Transición

Origen:
Approved

Destino:
Paid

Aceptar

Cancelar

Detalles

Agentes:
AccountUser
SuperAdmin

Servicio:
TPAY

Precondición:
Balance > 0 OR ps_ReturnAdvance = TRUE

Condición de control:

Mensaje en caso de Error:

Check the advanced money excess

Phase 7. Presentation Model.

Conjunto de Visualización

Nombre:

CV_Expense

▼

Limpiar

Borrar

Atributos a visualizar:

Atributo	Tipo dato
Project.ProjectName	String
Employee.EmpName	String
Employee.EmpSur...	String
Status	Int
AuthoDate	Date
PaymentDate	Date
TotExpenses	Real
Balance	Real

<< Añadir

Eliminar >>

Subir

Bajar

Agregar

Atributos:

Atributo	Tipo dato
Cause	String
AuthoDate	Date
AuthoComments	String
PaymentDate	Date
PayComments	String
TotExpenses	Real
TotExpensesCur	Real
Advances	Real
AdvancesCur	Real
Exchange	Real
Balance	Real
BalanceCur	Real

Clase: Expense

Aceptar

Cancelar

Phase 7. Presentation Model.

Filtro

flt_Expense

Alias

Expense Reports

Limpiar

Borrar

Fórmula

Project = vf_Project AND Employee = vf_Employee AND
PresentDate >= vf_DateInIssue AND PresentDate <= vf_DateEndIssue AND
AuthoDate >= vf_DateInApp AND AuthoDate <= vf_DateEndApp AND
PaymentDate >= vf_DateInPay AND PaymentDate <= vf_DateEndPay AND

<< Variable

Observ.

Variables

Nombre	Alias	Tipo dato	Tipo estilo	Estilo	Nueva	Modificar	Borrar
vf_Project	Project	Project	Sel. Población				
vf_Employee	Employee	Employee	Sel. Población				
vf_DateInIssue	Initial Issuing Date	Date					
vf_DateEndIssue	Final Issuing Date	Date					
vf_DateInApp	Initial Approving D...	Date					

Tipo

☒ Simple

☐ Objeto-valorado

Nombre

Alias

Tipo de dato

Estilo de introd.

Estilo de selección

Clase: Expense

Aceptar

Cancelar

Conceptual Model Validation

CARE Technologies, S.A.

Index

- Intro
- Overview
- Validation Degrees
 - Partial Validation
 - Total Validation

Index

- Validation Types
 - Elements of the Conceptual Model
 - Formulas of the Conceptual Model (Syntax)
- Validation Trees
 - Nodes
 - Leaves
- Example

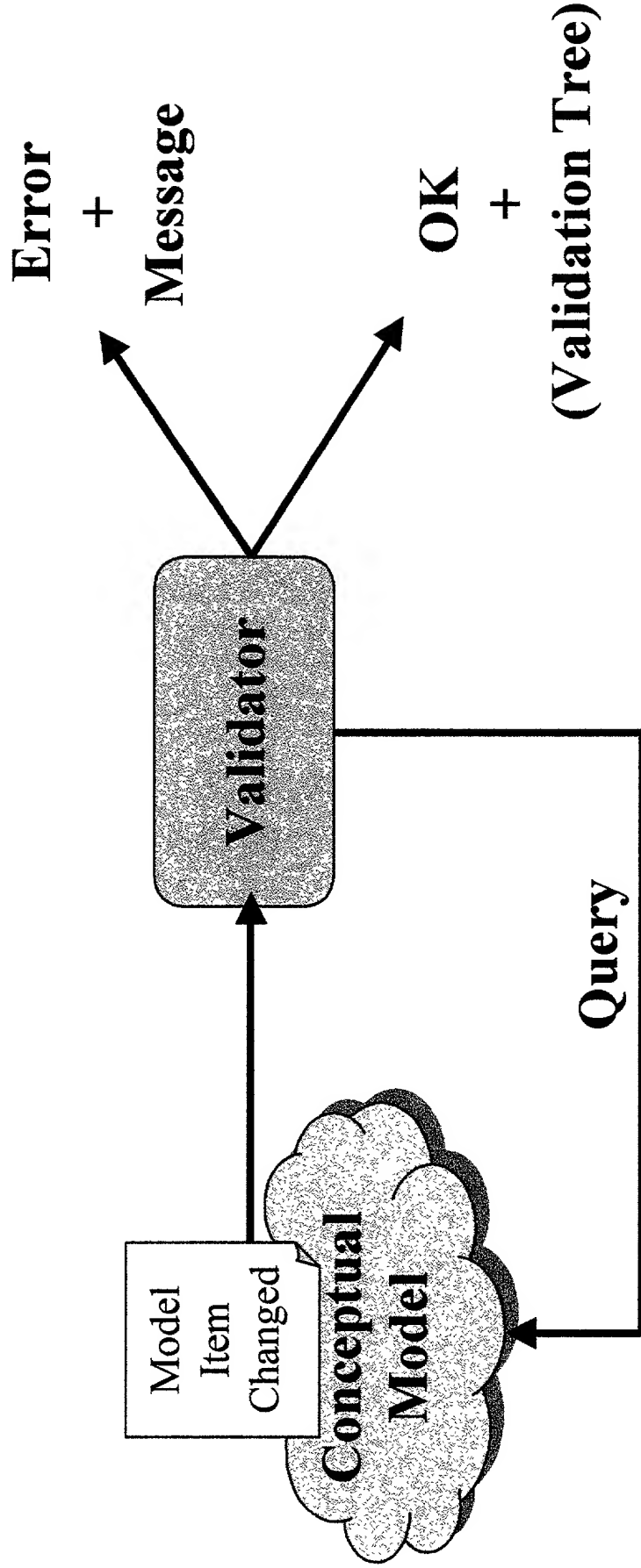
Intro

- Conceptual Model Validation is the process by which a conceptual model or a modification of it is proven to be valid:
 - Correct
 - Non Ambiguous
 - Non Contradictory
 - Complete
 - Every concept is fully specified
- Validation process checks the representation of requirements in Formal Specification Language to be valid

Validation Degrees

- Partial Validation
 - That of a single element of the Conceptual Model.
 - Happens whenever an element is added, modified or deleted.

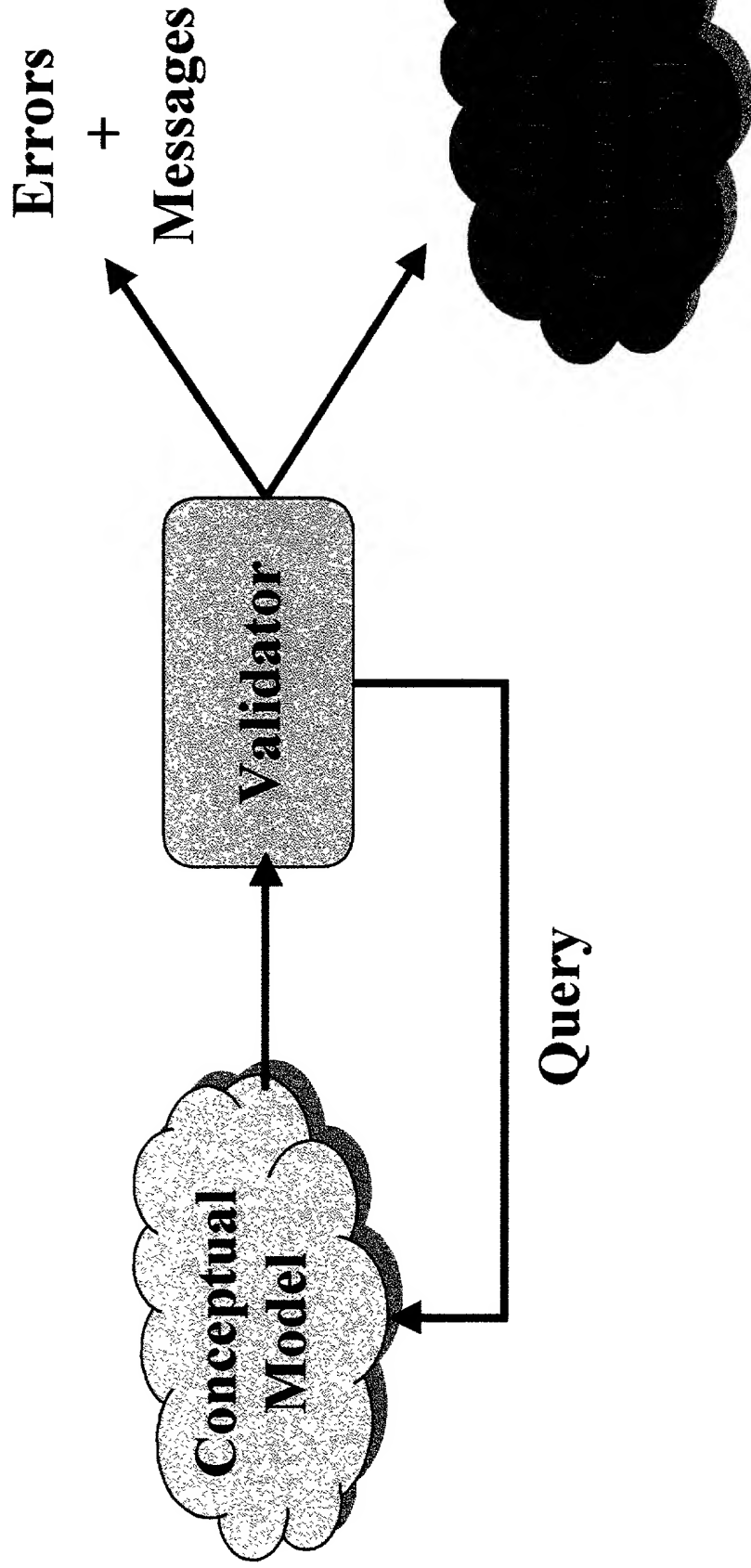
Partial Validation Overview



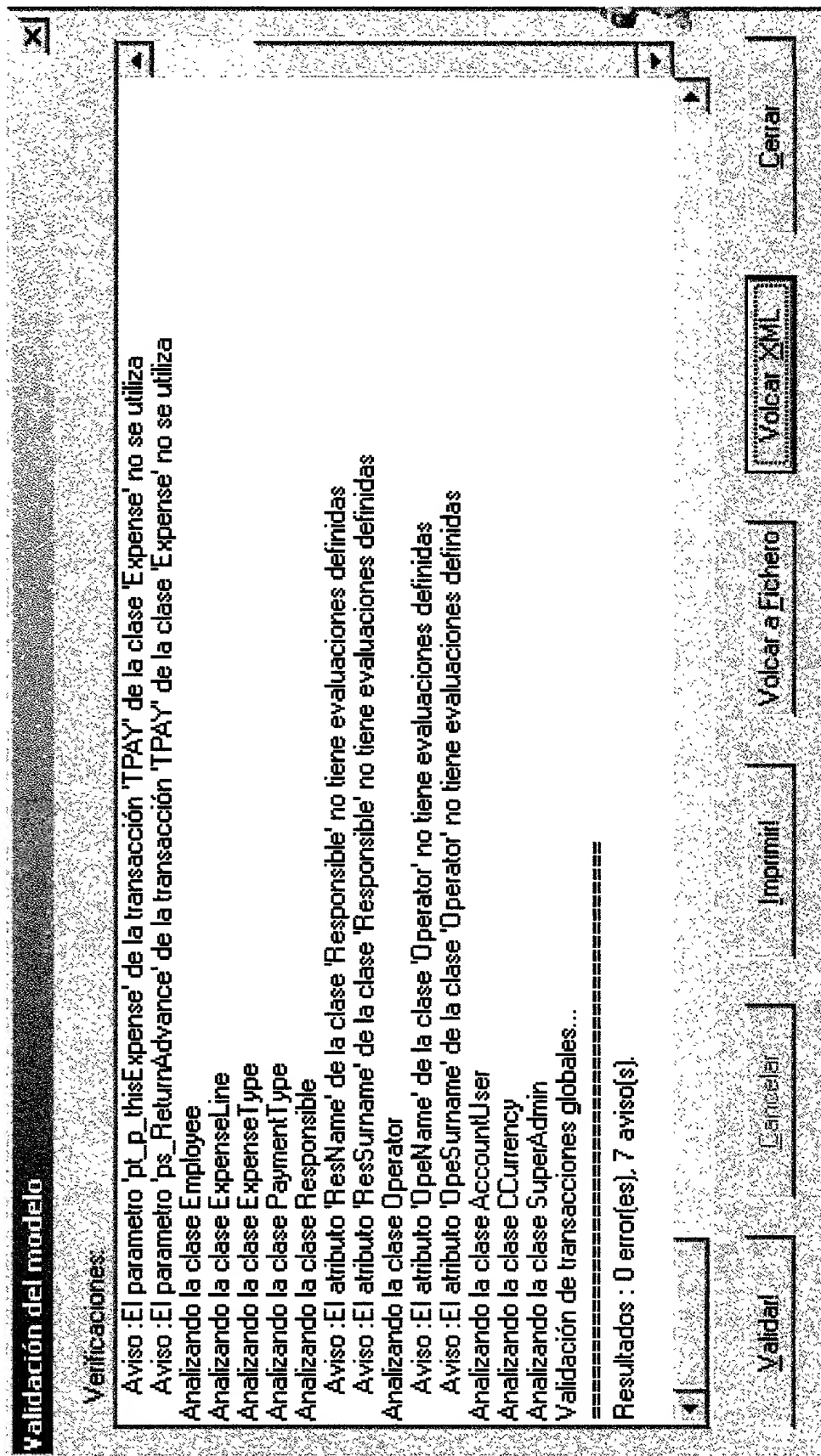
Validation Degrees

- Total Validation
 - That of the whole Conceptual Model.
 - Happens by request.
 - Must happen prior to any translation process.
 - Takes advantage of partial validations already performed.

Total Validation Overview



Total Validation Example



Validation Types

- Elements of the Conceptual Model
 - Ensure the properties of an element (except formulas) are correct and complete.
 - Conditions that must hold depend on the type of element and the property being validated.
 - Examples:
 - Class Name is unique in a Conceptual Model.
 - Attribute Name is unique in its Class (but not in a Conceptual Model)

Validation Types

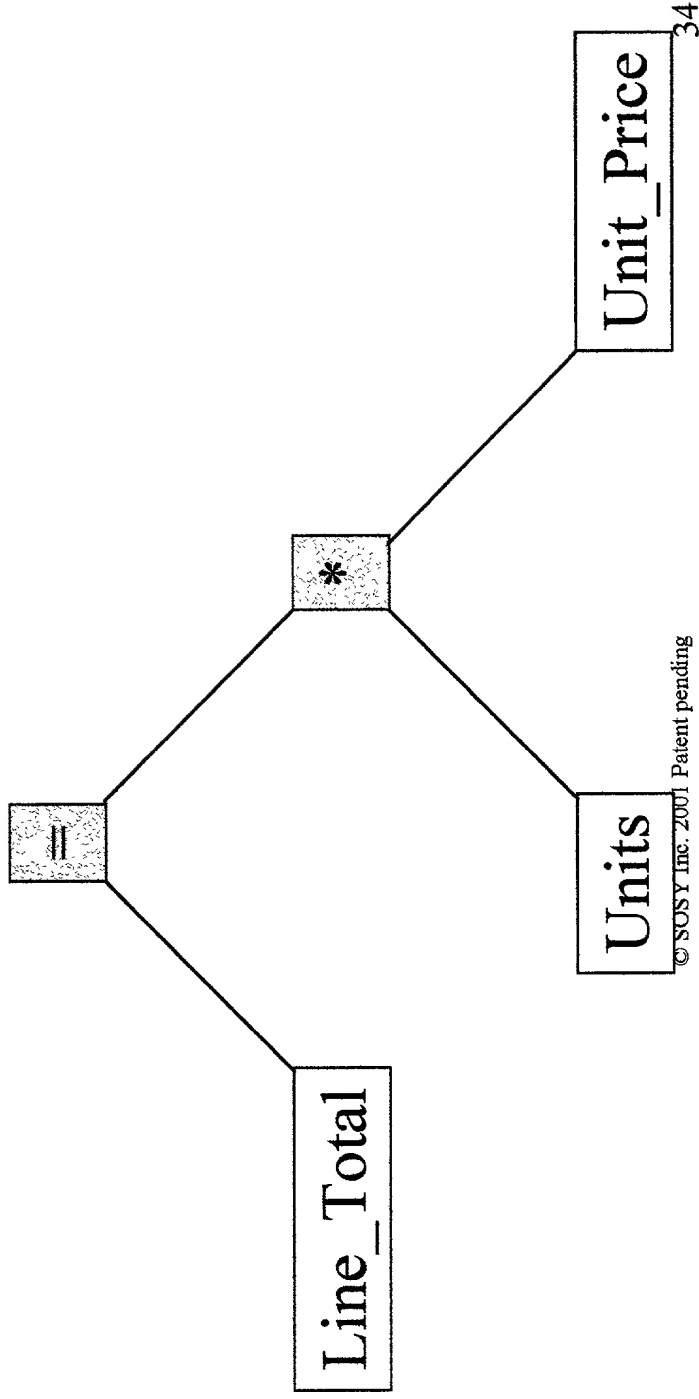
- Formulas of the Conceptual Model
 - Ensure the formulas of the Conceptual Model are correct and complete.
 - Syntactical and Semantical Validation according to an extended Formal Specification Language grammar.
- Input:
 - Formula expression
 - Formula Type (precondition, valuation, ...etc.)
 - Formula Context (class name, service name, ...etc.)
- Output:
 - Error Message (validation did not pass)
 - Validation Tree (validation passed)

Validation Trees

- Binary Tree representation of a correct formula.
- Tree consists of Nodes and Leaves.
- Nodes
 - Represent operators
 - Can have one or two “branches” (binary)
 - Branches can again be nodes or leaves
- Leaves
 - Represent operands
 - Have no branches

Example

- Line_Total = Units * Unit_Price



Documentation Translation

CARE Technologies, S.A.

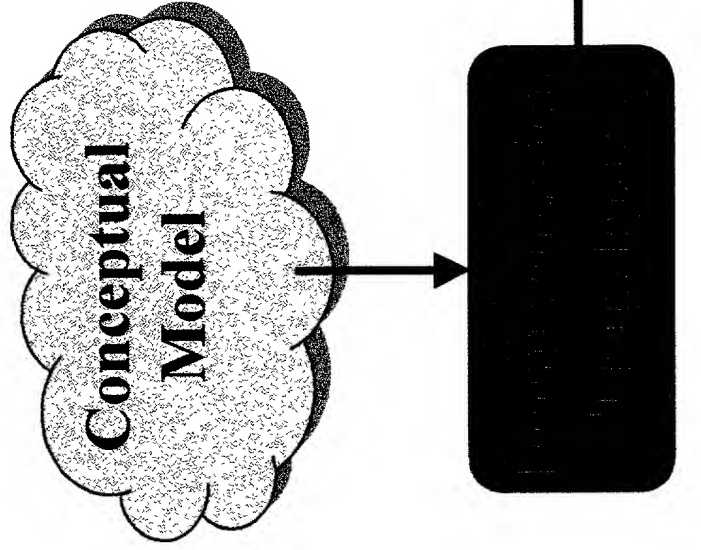
Index

- Intro
- Overview
- Output Detail
 - Document Types
 - Document Formats
- Translation
 - CM Subset of Interest
 - Translation Process
 - Remarks
- Example

Intro

- Documentation Translation is the process to obtain, from a Conceptual Model, documentation on the system it represents.
- Documentation can have several degrees of detail and be focused on different aspects, thus obtaining different documentation formats from the same Conceptual Model.

Overview



Document Type

- Help
- Full
- General
- User Help Manual
- Project Report
- Test Report

- Multifile HTML
- Single File HTML
- ASCII Text
- LaTeX
- RTF
- © SOSY, Inc. 2001. Patent pending. Compiled HTML

Document Format

Output Detail

- Document Types
 - Help
 - Description of each Class, its Attributes, Services and Population Selection Filters.
 - Full
 - Full description of a Conceptual Model
 - Aimed at analysts.
 - General
 - Description of each Class Attributes, Identification Function, Services, Aggregation Relationships and Specialization Relationships.

Output Detail

- Document Types
 - User Help Manual
 - Both Help Manual and Contextual Help (F1 key).
 - Intended for Operation Manual.
 - Integration with User Interface applications.
 - Project Report
 - Description of each Class Attributes and Services.
 - Test Report
 - Description of each Class Services.
 - Intended for Testing purposes.

Output Detail

- Document Formats
 - Multifile HTML
 - One HTML page per concept.
 - Recommended for navigable help.
 - Single File HTML
 - One single HTML page.
 - Recommended for printing.
 - ASCII Text
 - Single, plain ASCII text file.

Output Detail

- Document Formats
 - LaTeX
 - Single, LaTeX text file.
 - RTF
 - Single, RTF text file.
 - Compiled HTML
 - Same as Multifile HTML plus header files to be used by HTML Help Workshop compiler.
 - Recommended for contextual help.
 - Searching and Indexing facilities usage from browsers.

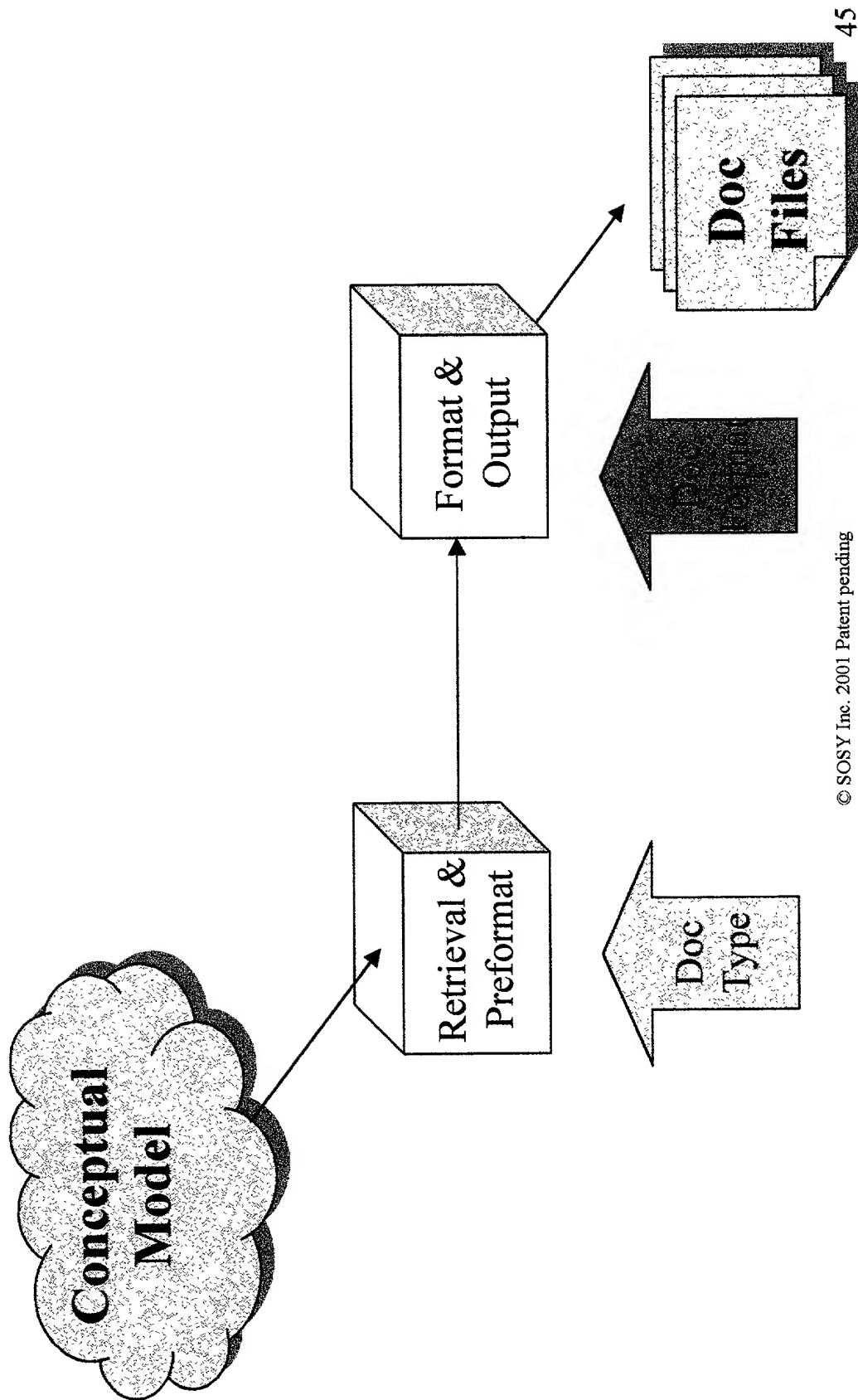
Translation

- Conceptual Model Subset of Interest
 - Subset of Interest depends on Document Type.
 - Usual elements:
 - Classes
 - Attributes
 - Relationships
 - Services & Arguments
 - Intensive use of analysis information.

Translation

- Translation Process
 - Read information from Conceptual Model and format it for output.
 - Two phases:
 - Information retrieval and pre-formatting.
 - Depends on Document Type
 - Independent from Document Format
 - Information output.
 - Depends on Document Format.
 - Independent from Document Type.

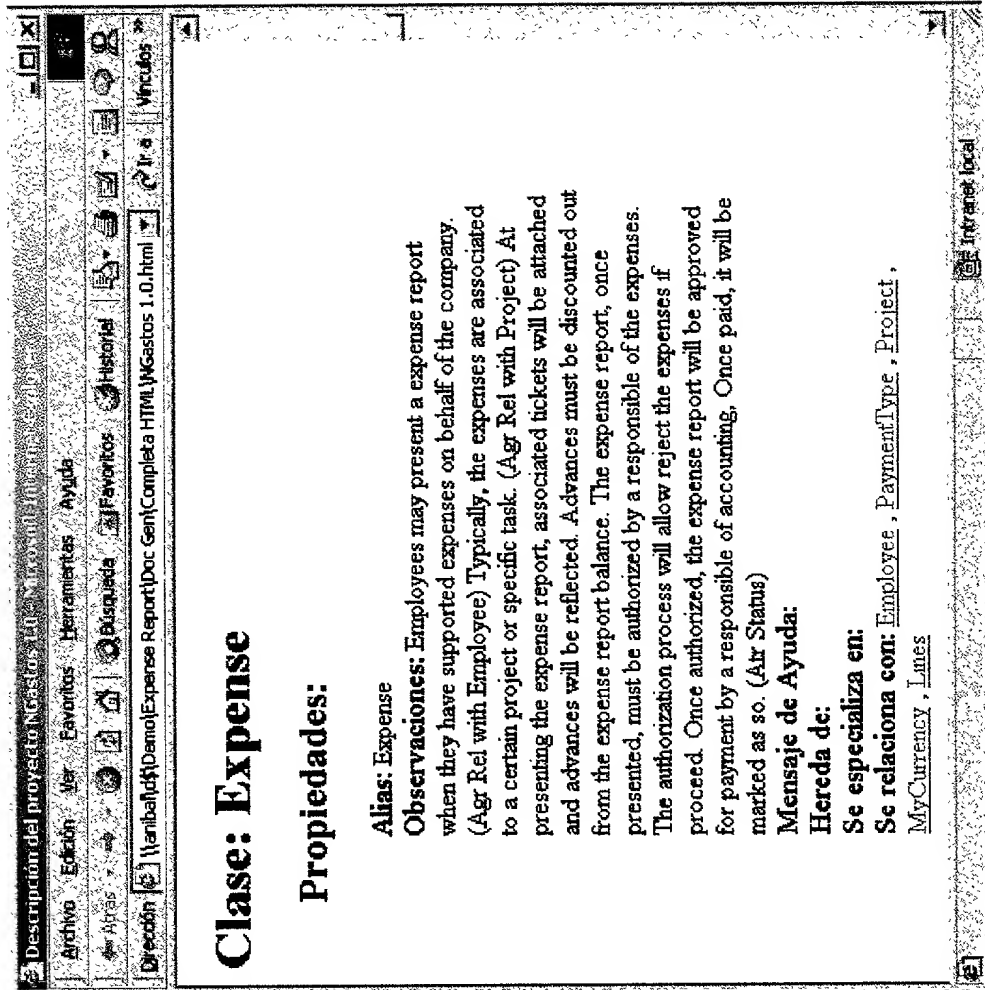
Translation Phases



Translation

- Remarks
 - Conceptual Model needs not to be valid (in terms of completeness and correctness) but it is always non-ambiguous.
 - The richer the analysis information, the richer the documentation.
 - Easily extensible
 - New Document Types
 - New Document Formats

Example



Persistence Relational Database Translation

CARE Technologies, S.A.

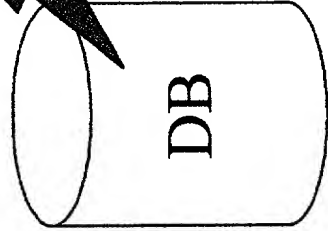
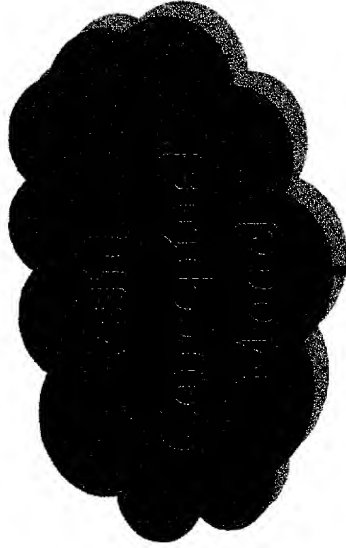
Index

- Intro
- Overview
- Output Detail
- Translation
 - CM Subset of Interest
 - Translation Processes
- Example

Intro

- Persistence Relational Database Translation is the process of creating a Relational Database from a certain subset of information in the Object Model of a valid Conceptual Model.
- Output script files are used to create a relational database using structured query language (SQL).

Overview



- Creates
- Primary Keys
- Foreign Keys
- Indexes
- Drop Creates
- Drop Primary Keys
- Drop Foreign Keys
- Drop Indexes

Output Detail

- **Creates**
 - Creation of Tables and Fields
- **Primary Keys**
 - Creation of Primary Keys as constraints on each table
- **Foreign Keys**
 - Creation of Foreign Keys as constraints on each table
- **Indexes**
 - Creation of Indexed on each table

Output Detail

- Drop Creates
 - Deletion of Tables
- Drop Primary Keys
 - Deletion of Primary Key Constraints
- Drop Foreign Keys
 - Deletion of Foreign Key Constraints
- Drop Indexes
 - Deletion of Indexes

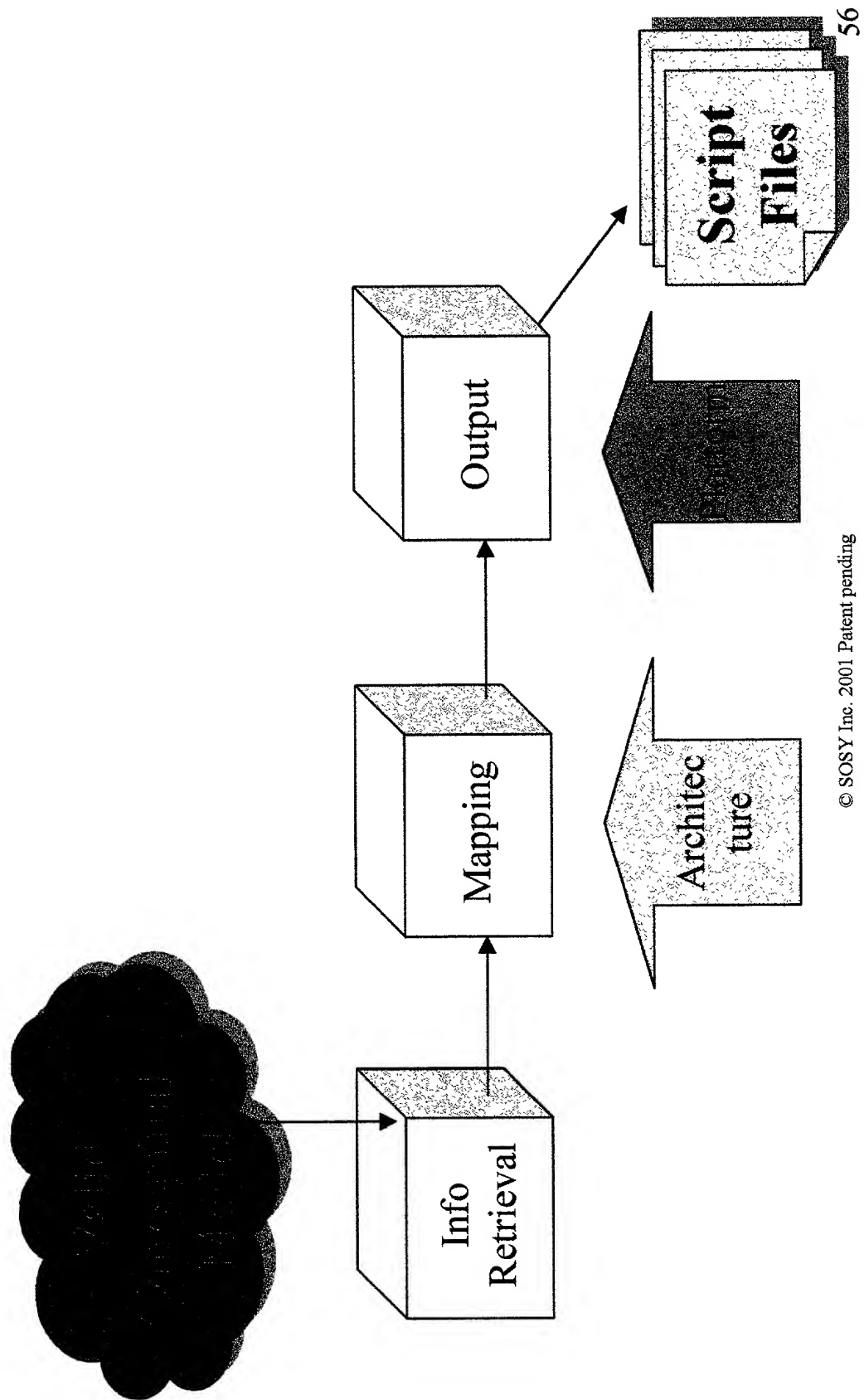
Translation

- Conceptual Model Subset of Interest
 - Object Model
 - Classes
 - Attributes
 - Identification Functions
 - Aggregation Relationships
 - Inheritance Relationships

Translation

- Three phases:
 - Information retrieval.
 - Independent from persistence architecture.
 - Fixed architecture mapping.
 - Depends on persistence architecture.
 - Information output.
 - Targeted for Standard ANSI SQL 92 RDBMS.
 - Script files depends on the platform's SQL syntax of RDBMS manufacturer.
 - May depend on platform specifications to make use of manufacturer extensions and tuning.

Translation Phases



Translation

- Translation Processes. Mapping:
 - Class \rightarrow Table
 - Non-derived Attribute \rightarrow Field
 - Identification Function \rightarrow Primary Key
 - Univaluated Relationship \rightarrow Foreign Key
 - Univaluated Relationship \rightarrow Index
 - Multivaluated Relationship \rightarrow Table
 - Inheritance Relationship \rightarrow Foreign Key

Example

Create table script in SQL for Expense class

```
CREATE TABLE Expense (
    fk_Project_1 int NOT NULL ,
    id_Expense int NOT NULL ,
    fk_Employee_1 CHAR(10) NOT NULL ,
    fk_MyCurrency_1 CHAR(5) NOT NULL ,
    fk_PaymentType_1 CHAR(5) NULL ,
    PresentDate datetime NOT NULL ,
    Status int NOT NULL ,
    Cause VARCHAR(255) NOT NULL ,
    AuthoDate datetime NULL ,
    AuthoComments VARCHAR(255) NULL ,
    PaymentDate datetime NULL ,
    PayComments VARCHAR(255) NULL ,
    Advances DECIMAL(19,6) NOT NULL ,
    Exchange DECIMAL(19,6) NOT NULL);
```

Business Logic Translation

CARE Technologies, S.A.

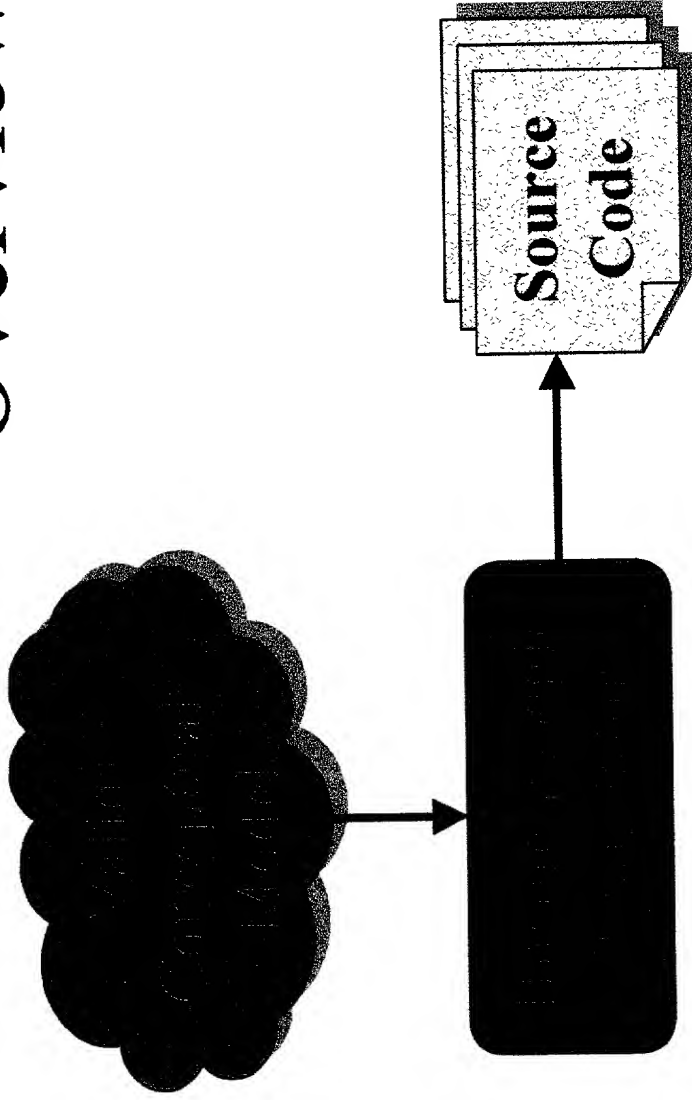
Index

- Intro
- Overview
- Output Detail
- Translation
 - CM Subset of Interest
 - Translation Processes
- Example

Intro

- Business Logic Translation is the process to obtain, following a precise Execution Model, the source code corresponding to the business logic from a valid Conceptual Model for a target Programming Language and Software Architecture.
- Execution Model is independent from Programming Language and Software Architecture.

Overview



Determines:

-Target Programming Language

-Target Software Architecture

Output Detail

- Target Programming Language and Software Architecture determine:
 - Source code organization in files
 - Files internal organization
- Source Code's backbone: Execution Model.

Output Detail

- Traceability: Source code highly readable and maintainable thanks to:
 - Source code is always organized and structured in the same way.
 - Naming conventions applied.
 - Source code includes analysis information from the Conceptual Model as comments.

Output Detail

- Implementation of a precise Execution Model grants Functional Equivalence with Conceptual Model.
- Programming Interface to Clients for:
 - Actor Validation and Authentication.
 - Services Execution.
 - Queries Execution.
- Manages:
 - Concurrency.
 - Transactions.
 - Interoperable Objects Persistence.

Translation

- Conceptual Model Subset of Interest
 - Object Model
 - Static properties (Visibility & Persistence)
 - Attributes + Identification Functions
 - Derivations
 - Aggregation Relationships
 - Inheritance Relationships
 - Services (Execution Model)
 - Arguments
 - Preconditions
 - Transaction Formulas
 - Actors (Execution Model)
 - Integrity Constraints (Execution Model)

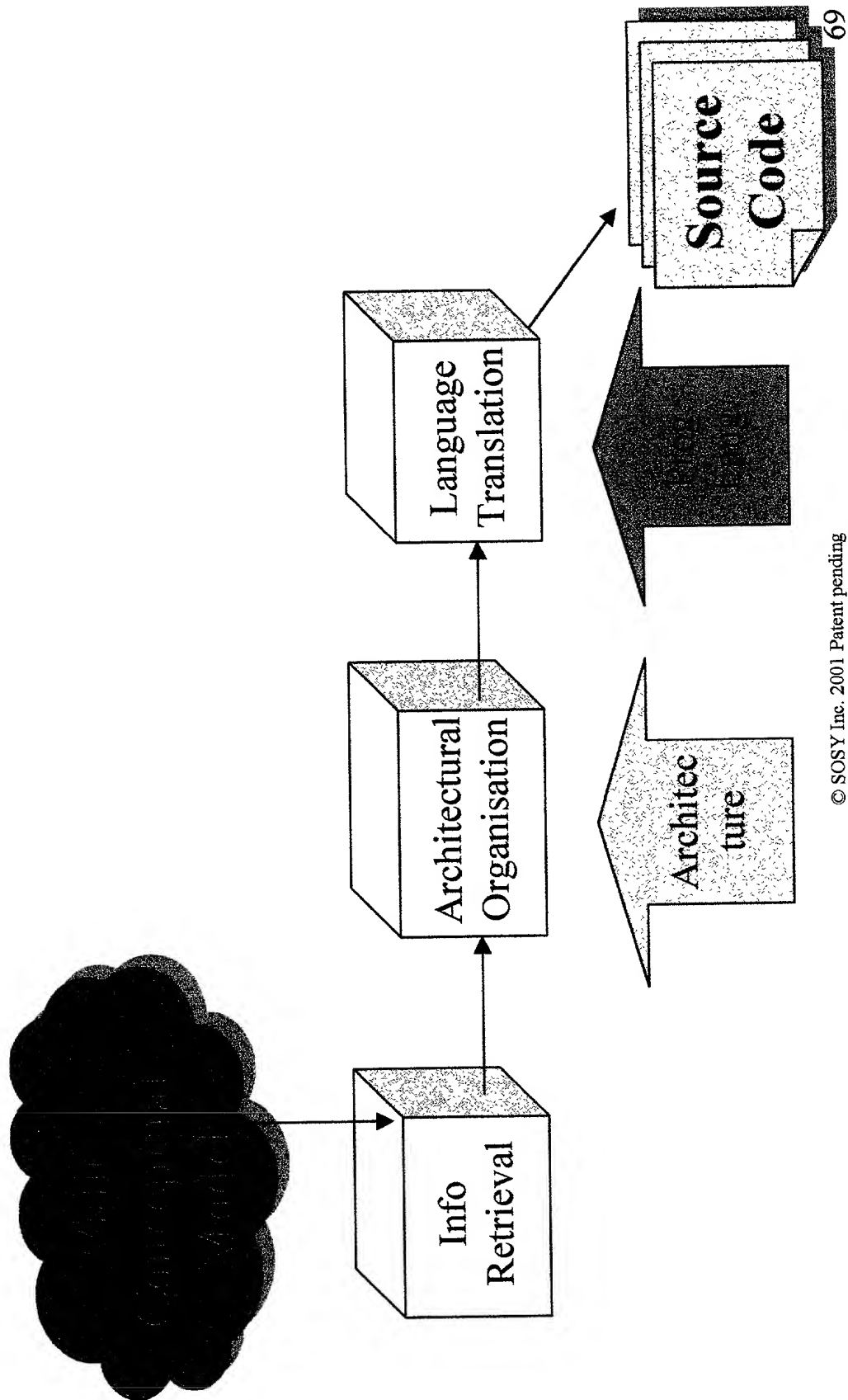
Translation

- Conceptual Model Subset of Interest.
 - Dynamic Model.
 - State Transition Diagram (Execution Model).
 - Controls Valid Lifes for an Object.
 - Object Interaction Diagram.
 - Triggers (Execution Model).
 - Global Transactions (Execution Model).
 - Functional Model (Execution Model).
 - Object state change upon occurrence of an event.

Translation

- Translation phases:
 - Information retrieval
 - Independent from target Software Architecture and Programming Language
 - Architectural organisation
 - Depends on target Software Architecture
 - Independent from target Programming Language
 - Determines files organisation and files internal structure
 - Language translation
 - Depends on target Programming Language
 - Influenced by Software Architecture
 - Takes advantage of Programming Language capabilities

Translation Phases



Translation

- Translation Processes
 - Classes
 - Static properties translation
 - Services translation
 - Queries translation
 - Global Interactions
 - Services translation
 - Global Functions
 - Functions Interface translation
 - Body is left blank

Example

- Evaluation:
 - Service Authorize modifies attributes Status, AuthoDate and AuthoComments
 - Formal Specification Language expression for evaluation Valuation
[authorize ()] Status=2 and AuthoDate=today() and AuthoComments="" ;

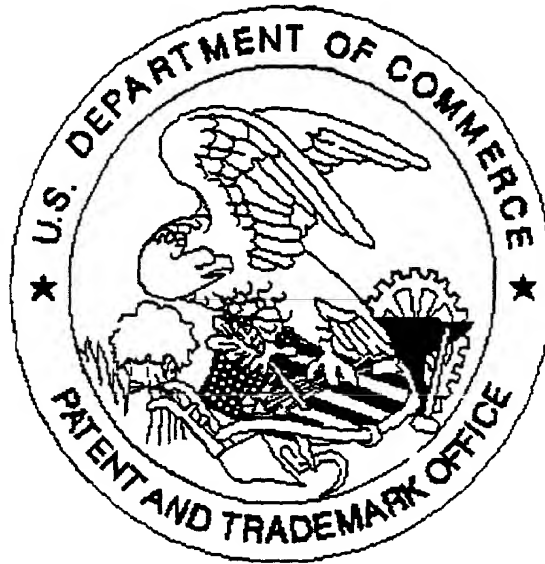
- Visual Basic Produced

```
Private Function MV_Eval_Expense_authorize() As String
    Expense_Status = 2
    Expense_AuthoDate = today()
    Expense_AuthoComments = ""
    MV_Eval_Expense_authorize = ""
End Function
```

User Interface Translation

CARE Technologies, S.A.

United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☒ Scanned copy is best available. Drawings and miscellaneous